MEC Software

BarPump™

Technical Manual

Version 1.6 (Beta Test)

© Copyright 2002, 2003 Measurement Equipment Corporation
All Rights Reserved

BarPump
Technical Manual

© Copyright 2004 Measurement Equipment Corporation
All Rights Reserved

Distributed by:

The Barcode Software Center
1113 Hull Terrace
Evanston, Illinois USA 60202

Tel. 847-866-7940
Fax. 847-866-9836
www.makebarcode.com

Introduction

BarPump is designed to add barcodes reports printed on laser printers which are compatible with the PCL-5 printer command language.

Print data is sent to *BarPump* instead of directly to a printer. *BarPump* extracts data marked by trigger commands and replaces it with codes that will print the data as a barcode. All other data is unaffected; the appearance of the final document is the same as the original save for the addition of barcodes. *BarPump's* output can be sent directly to a printer port or to another file.

Processing of the printer stream is quite fast. *BarPump* uses MEC Software's font generator technology to create precise barcodes quickly and with very little data overhead: only 3 to 6 kilobytes of overhead for a typical print job plus another 10 to 20 characters per barcode printed. The low overhead translates into quicker printing and lower impact on network traffic, especially when com pared with graphic-based barcode printing solutions.

Notes for the Beta Version of BarPump

As of this writing, not all of the features documented in this manual are supported. This advance Beta release intended for evaluation of the suitability of BarPump for particular applications. Here is a short list of the currently-available features; others are being regularly added, so check with us to see if a feature you require has been recently implemented:

- Process input file to an output file in batch mode
- Process input file to installed (named) Windows printer in batch mode
- Process files from multiple directories to multiple installed printers in server mode
- All standard linear barcode types shown in the manual

Quick Start Guide

Process a File in Batch Mode

BarPump recognizes the same trigger commands used by popular types of printer add-in hardware modules. If your system uses these modules, you can test BarPump by sending a sample report to a file instead of a printer; name the file something like testreport.txt. Next, process the file with BarPump (we are using the demonstration version of BarPump here, barpumpd.exe). *Note that the path to the program file is enclosed in quotations because it contains a space character (the Program Files folder); the quotes prevent Windows from getting confused.*

```
C:\> "C:\Program Files\BarPump\barpumpd.exe" testreport.txt outfile.txt
```

Processing requires only a moment. Next, copy the output file to a printer that does not have a barcode module installed. *We are using the /B (binary) option to prevent Windows from fiddling with the content of the file.*

```
C:\> copy /B outfile.txt LPT1:
```

The result should be a report complete with barcodes, identical to the reports you normally create but generated on a printer with no installed barcode hardware module.

Process Multiple Files in Server Mode

BarPump can monitor a disk folder and wait for files to appear, then automatically process them and forward them to a printer. A system can print barcoded reports on any laser printer simply by saving the report file to a specific directory.

For our example, let's assume that we have two laser printers installed on the system ("HP4000TN" and "MainOfficeLaser"). These printers do not need to be directly attached to our computer; as long as they appear on our list of installed printers, BarPump can print to them.

First, create the directories that will act as "mail boxes" where new reports can be dropped off. The directories can be on the local hard disk or on a shared network drive. To make things convenient, we'll to mimic the installed names of the printers and create C:\TempReport\HP4000TN and C:\TempReport\MainOfficeLaser.

We need to tell BarPump about these directories and printers by editing the configuration file. Use Notepad or any text editor and open C:\Program Files\BarPump\BarPump.cfg and add FORWARD statements. The general form of the FORWARD statement is:

FORWARD= <directory to watch> , <destination printer>

Here are the statements for our example (lines beginning with # are comments):

```
#
#     BarPump.cfg
#
FORWARD=C:\TempReport\HP4000TN\*.*,HP4000TN
FORWARD=C:\TempReport\MainOfficeLaser\*.*,MainOfficeLaser
```

Now run BarPump

```
C:\> C:\Program Files\BarPump\barpumpd.exe
```

BarPump will read the default configuration file BarPump.cfg and set itself up according to the FORWARD statements. Leave BarPump running and test it by copying Test1.txt from C:\Program Files\BarPump and pasting it into C:\TempReport\HP4000TN. In a moment, BarPump will automatically process the file and send it to the HP4000TN printer. Paste the same file into C:\TempReport\MainOfficeLaser to print it on the Main Office Laser.

Running BarPump as a Service

We are working on this feature, which will run BarPump as an “always on” background service.

Command Line Arguments

Specifying Input and Output

In its simplest form, the *BarPump* command line requires only the name of the program, the source document file, and the destination where the finished output should should be written. The source (input) must precede the destination (output). For example:

```
C:\> barpump rawprint.txt barcodes.txt
```

BarPump will read the file rawprint.txt, insert barcodes, and write the result to the file barcodes.txt. If either the source or destination name contains spaces, the name must be enclosed in quotation marks. For example:

```
C:\> barpump “july sales report.txt” barcodes.txt
```

Files are presumed to be in the default directory unless a full pathname is specified. For example:

```
C:\> barpump \documents\salesreport.txt \temp\barcodes.txt
```

The input and output can be explicitly identified with the -i (input) or -o (output) command line option. If the options are used, arguments can appear on the command line in any order. Here is the first example shown with the explicit options:

```
C:\> barpump -irawprint.txt -obarcodes.txt
```

BarPump can work with a variety of data sources and destinations. See the section on Source and Destination Prefixes later in this section.

Specifying the Configuration File

By default, *BarPump* will look for the configuration file barpump.cfg in the same directory where the *BarPump* executable file is stored. If the configuration file cannot be found, *BarPump* will function properly but with minimum features. If it is necessary or desirable to store the configuration file in a different location, or to have several configuration files for different situations, the pathname of the configuration file may be specified on the command line with the –c option:

```
C:\> barpump rawprint.txt barcodes.txt –c/usr/special.cfg
```

The –c option may appear anywhere on the command line; the pathname must immediately follow the –c with no intervening spaces.

Specyfing Batch or Server Mode

BarPump has two basic modes of operation: Batch Mode and Server Mode. In Batch Mode, data is read from the source until the end is reached; the finished result is sent to the destination and *BarPump* exits. In Server Mode, after completing processing of a document *BarPump* waits for more documents and automatically processes each one as it arrives. The default mode of operation depends on the type of input (see the following section). The default mode can be overridden with the -mBATCH or -mSERVER command line option.

```
C:\> barpump -mBATCH IP:223.122.100.005:48 LPT1:
```

Source and Destination Prefixes

BarPump can read from and write to more than just files. The type of source or destination is specified on the command line through the use of prefixes.

File

The FILE: prefix explicitly defines the source or destination as a file. In the absence of any recognized prefix, this is the default type. For example, the following command lines have the same effect:

```
C:\> barpump salesreport.txt barcodes.txt
C:\> barpump FILE:salesreport.txt FILE:barcodes.txt
```

Batch Mode is the default for FILE inputs.

Local Parallel Port

The name of a local parallel port in the form LPT*n*: when *n* is the port number will direct output directly to the port. Parallel ports are valid for destinations only.

```
C:\> barpump salesreport.txt LPT1:
```

Local Serial Port

Data can be received from or sent to a local serial port by specifying the port number. The communications parameters are optional; if they are not specified, BarPump will use the current default settings for the port.

COM*n*:[<baud>,<parity>,<data bits>,<stop bits>]

```
C:\> barpump salesreport.txt COM1:9600,N,8,1
```

Server Mode is the default for COM inputs

Standard Input and Standard Output

Standard Input and Output refer to data passed directly from the operating system to a program through re-direction or the pipe operator. This permits the output of one program to be sent directly to the input of another. Just use STDIN or STDOUT to specify the source or destination. In the following example, we assume that the program makereport generates a report; the output of this program is directed to BarPump which in turn sends its output to a printer port:

```
C:\> makereport > barpump STDIN LPT1:
```

Batch Mode is the default for STDIN inputs.

Named Pipe

Named Pipes are data queues maintained by the operating system that permit passing data from one program to another. An advantage of pipes is that data can be sent to the pipe from multiple sources; the pipe keeps everything in order and the program which is reading the pipe can deal with each block of data in the order in which it was received. Use the PIPE prefix for input or output. In this example, print jobs are taken from the reportqueue pipe and sent to parallel port LPT1:

```
C:\> barpump PIPE:reportqueue LPT1:
```

Server Mode is the default for PIPE inputs

IP Address

Use the IP prefix followed by the IP address and Port Number to use internet protocol to receive or send data. The port number must be specified. The IP address may be omitted for inputs, in which case the system IP address is assumed (the port number must still be specified). Here are some examples:

```
C:\> barpump IP:223.140.100.104:1045 LPT1:
C:\> barpump IP:1045 IP:223.140.100.006:1446
C:\> barpump report.txt IP:223.140.100.006:1446
```

Server Mode is the default for IP inputs.

FTP (file transfer protocol)

Similar to IP, FTP uses a pre-defined port number and protocol to transfer data. Use the FTP: prefix and specify the IP address; if the IP address is omitted for the input, the system's IP setting is assumed.

```
C:\> barpump FTP:223.140.100.104 LPT1:
C:\> barpump FTP: FTP:223.140.100.006
C:\> barpump report.txt FTP:223.140.100.006
```

Server Mode is the default for FTP inputs.

Directory

The DIR: prefix is used to specify a particular directory to be watched for the arrival of files. BarPump will detect the arrival of files and process them automatically. DIR: works with inputs only and should always be specified with an asterisk wildcard:

```
C:\> barpump DIR:\temp\printdocs\*.* LPT1:
C:\> barpump DIR:\temp\printdocs\*.txt LPT1:
```

Server Mode is the default for DIR: inputs.

Named Printers

The PRN: prefix indicates that the output is a named printer. The second of these two examples uses a printer with spaces in its name, so the name must be enclosed in quotation marks:

```
C:\> barpump report.txt PRN:MainPrinter
C:\> barpump report.txt "PRN:Main Report Printer"
```

Barcode Triggers

A trigger is a specific command or character pattern in the printer data that helps *BarPump* determine where barcodes should be inserted. *BarPump* responds to several types of triggers:

BarSIMM/BarDIMM

The JetCAPS BarSIMM and BarDIMM are plug-in hardware modules that add intelligent barcode capability to Hewlett-Packard printers. A compatible module is available from Lexmark for use in their printers. These modules use a PCL-5 font selection syntax which is supported by SAP R/3 systems. *BarPump* will respond to these commands and produce the same printed result, limited to the particular barcode types supported by *BarPump*.

The BarSIMM/BarDIMM escape sequence contains all of the information needed to construct a barcode: type of barcode, height, width of bars, etc. The configuration file required by other types of triggers is unnecessary with BarSIMM/BarDIMM syntax. Details on the syntax of this type of command are provided in an appendix to this manual.

Generic [Start] … [End] Triggers

A number of barcode programs have, over the years, used a variety of special character patterns to mark the start and end of barcodes. For example, Bear Rock Software (now departed) used @@ to mark the start and end of a barcode. The following string, embedded in the print image file, would produce a barcode containing “12345”:

```
Invoice No. @@12345@@
```

This type of trigger marks the data to be encoded in the barcode, but does not provide any information about the nature of the barcode itself (type of code, height, bar width, ratio, etc.). *BarPump* uses a configuration file to specify details about the barcode that should be printed in response to a particular trigger. For example, each of the following triggers could produce a different type or size of barcode.

```
Job No. <BARCODE>33398</BARCODE>
Customer No. [BC1]440958[XBC]
```

Any number of [Start] … [End] patterns can be specified in the configuration file, each with its own particular barcode characteristics. This makes it possible to respond to triggers used by legacy software or to create entirely new trigger patterns. The “end of barcode” marker may be a specific character pattern, or it may be specified as a carriage return, line feed, or any non-printable control code.

Data Pattern Triggers

In some cases, existing software may not include any trigger sequences in the printer data and it may not be possible to modify the software. Through the configuration file *BarPump* can be configured to recognize specific data patterns and respond by inserting a barcode.

higher resolutions automatically scale the barcodes, so the size of the barcode remains the same regardless of the resolution of the printer. Printers with 600 dpi permit a greater selection of barcode widths (see the NARROWBAR statement later in the manual) and can be helpful especially when printing small barcodes. *Use 600 dpi only if you are absolutely certain that ALL target printers have 600 dpi or higher resolution.*

There are only two resolutions allowed: 300 (the default) or 600; any other resolution is not allowed under the PCL-5 specification and will cause BarPump to use the default 300 dpi setting.

Named Sections

The configuration file may contain any number of named sections, each of which begins with a Section statement and ends with an End Section statement: The name of each section must be unique.

```
Section <name of section>
      |
      <section parameter statements>
      |
End Section
```

Statements within each section define the trigger and the characteristics of the barcode that will be printed.

Section Parameter Statements

The following statements define the character patterns that trigger printing of a barcode. The trigger pattern, data, and terminator must all appear on the same line in the print image data. Each Section should have a TRIGGER, DATACOL/DATAROW, or a DATAPATTERN statement (choose one only). Each section should also include a TERMINATOR statement.

TRIGGER

```
TRIGGER=<character string>
```

This statement defines a fixed character pattern that will immediately precede the data to be printed as a barcode. Here are some examples

```
TRIGGER=@@
TRIGGER=BARCODE1
TRIGGER=BARCODE2
```

Triggers should be sufficiently peculiar so that there is little or no risk of normal data accidentally duplicating a trigger sequence. For example, START would be a bad trigger since the word “Start” might appear in a document as normal data; it would be better to use XXXSTARTXXX.

DATACOL and DATAROW

For example, an invoice printout may include the phrase “Invoice No.“ on every page just before the invoice number. BarPump can locate the phrase “Invoice No.” and print the data immediately following as a barcode. The barcode can be printed instead of the original data or in addition to it.

For reports that do not contain (and cannot be made to contain) trigger codes, *BarPump* can identify data by its column and row position on a report. This works best for reports which are printed in plain line printer style. The column location of the data must be a single value; the row may be a single row or a range of rows. This feature can be useful for printing barcodes in multi-line reports. For example:

```
DATACOL=15
DATAROW=20-30
```

When using a range of rows it is recommended that DATALENGTH, DATAPATTERN, TRIMLEFT, and TRIMRIGHT be used to suppress printing of barcodes for lines which are empty.

DATALENGTH

```
DATALENGTH=<number of characters>
```

This statement sets limits on the length of the data string that will be printed as a barcode. A single number specifies a fixed length, while a range specifies a minimum and maximum length. For example:

```
DATALENGTH=15
DATALENGTH=1-15
```

If the length of the actual data string is less than the minimum, the barcode will not be printed. If the length of the actual data is zero and the minimum allowed length is zero, the barcode will be printed but with no data content.

DEMOTEXT
DATAPATTERN

```
DEMOTEXT=<character string>
DATAPATTERN=<character search string>
```

This statement uses a pattern-matching string to find the desired data in the print image file.

When operating in demonstration mode, BarPump prints the phrase “BarPump Demo” on the face of each barcode. The character string specified by the DEMOTEXT statement is printed below “BarPump Demo” on each barcode. For example,

TERMINATOR

```
DEMOTEXT=Acme Software Shop
TERMINATOR=<character string>
```

This statement will cause the following to be printed on the face of each barcode while BarPump is running in demonstration mode:

This statement defines the characters that mark the end of the data which is to be printed as a barcode. The terminator may be a specific character string, one of several pre-defined control codes, any non-printable control code, or any character not included in the character set of the particular barcode type being used.

```
BarPump Demo
Acme Software Shop
```

RESOLUTION

```
TERMINATOR=@@
```

Any of the following special terminators may be specified:

```
RESOLUTION=<target printer resolution in dpi>
```

CR	(carriage return)
LF	(line feed)

BarPump presumes a printer resolution of 300 dpi when creating the barcodes. Printers with

The Configuration File

The *BarPump* configuration file is a plain ASCII text file which conforms to the following syntax guidelines:

- Indenting can improve readability. Leading spaces or tabs are ignored by *BarPump* and so have no effect on the output.
- Comments may be included by inserting a # character in the leftmost column. Any line beginning with # is ignored by *BarPump*.
- Non-printable control codes may be included in strings by enclosing the decimal ASCII value of the character within square brackets. For example, [27] represents the escape character (ASCII decimal value of 27).

The configuration file is divided into sections. The Header Section comes first and defines parameters which apply to the general operation of *BarPump*. There can be any number of named sections following the header section. Each section is identified by a name at its beginning, and its end is marked by an End Section statement.

Header Section

The header section is presumed to begin with the start of the configuration file and end with the beginning of the first named section or the end of the configuration file (assuming there are no named sections).

CRLF	(carriage return or line feed)
NONPRINT	(any non-printable control code)
NOTINSET	(any character not supported by barcode)

STRIPTRIGGER and **STRIPTERMINATOR**

```
STRIPTRIGGER=<YES/NO>
STRIPTERMINATOR=<YES/NO>
```

These statements specify if the trigger and terminator should be removed from the data before it is sent to the printer (the default is yes, remove them). Specific trigger strings like @@ or <BARCODE> are typically removed. Use this statement if the trigger or terminator should be included in the finished report.

For example, let's say that adding a specific trigger like @@ to the data is difficult but that the phrase "Invoice No." always comes right before the piece of data that should be barcoded (and does not appear anywhere else in the report). Let's also assume that the invoice number is the last piece of data on the line and is followed by a carriage return.

```
SECTION INVOICE
     TRIGGER=Invoice No.
     STRIPTRIGGER=NO
     TERMINATOR=CR
     STRIPTERMINATOR=NO
     SYMBOL=CODE39
ENDSECTION
```

If the document contains "Invoice No. 12345", the finished document will contain the text phrase "Invoice No." followed by 12345 printed as a barcode. We may wish to leave the entire "Invoice No. 12345" in the finished document and print the barcode somewhere else on the page. We can add the following statements to leave the original data in place and print the barcode 1 inch to the right and 2.5 inches down measured from the left end of "12345":

```
SECTION INVOICE
     TRIGGER=Invoice No.
     STRIPTRIGGER=NO
     TERMINATOR=CR
     STRIPTERMINATOR=NO
     SYMBOL=CODE39
     REPLACEDATA=NO
     HORPOS=+1.0
     VERPOS=+2.5
ENDSECTION
```

See the sections on REPLACEDATA, HORPOS, and VERPOS for details.

TRIMLEFT and **TRIMRIGHT**

```
TRIMLEFT=<YES/NO>
TRIMRIGHT=<YES/NO>
```

These statements cause *BarPump* to remove leading or trailing spaces from the data before printing it as a barcode. For example,

```
TRIMRIGHT=YES
```

will cause trailing spaces to be dropped from the data. Here are the ways the sample trigger and data @@ 12345 @@ will be affected. The result string is shown with quotation marks for clarity.

Trimleft	Trimright	Result
NO	NO	“ 12345 “
NO	YES	“ 12345”
YES	NO	“12345 “
YES	YES	“12345”

SHOWASTERISKS

```
SHOWASTERISKS=<YES/NO>
```

Many methods of printing Code 39 barcodes require that the data be enclosed in asterisks (*), which by common practice translate to the start/stop character that must begin and end every Code 39 barcode. BarPump inserts the start/stop codes in the barcode automatically, and there is no requirement for the asterisks to appear in the readable text. Use this statement to control whether the asterisks appear or not according to your preference. The data content of the barcode is unaffected.

REPLACEDATA

```
REPLACEDATA=<YES/NO>
```

This statement specifies if the barcode will replace the variable data in the print image file or if the barcode will be printing in addition to the original data.

```
REPLACEDATA=YES
```

This statement will cause BarPump to remove the variable data from the print image file and replace it with the barcode (this is the default behavior). On the other hand,

```
REPLACEDATA=NO
```

Tells BarPump to pass the original data through to the printer and print the barcode as well. To avoid printing the barcode on top of the original text, see the HORPOS and VERPOS statements.

HORPOS and **VERPOS**

```
HORPOS=<horizontal position in inches>
VERPOS=<vertical position in inches>
```

These statements specify where the barcode will be printed. If these statements are omitted, the barcode will be printed where the text in the original print image file would have been

printed. It may be desirable to move the barcode to a new location, especially if both the barcode and the original data are being printed (REPLACEDATA=NO).

The position of the barcode is measured from its bottom left corner, and the dimension is specified in inches. A dimension with no sign (no plus or minus) is taken as an absolute position measured from the top left corner of the page. The following example prints the barcode 5.50 inches down and 3.25 inches right as measured from the top left corner of the page:

```
VERPOS=5.50
HORPOS=3.25
```

A plus sign indicates right or down, and a minus sign indicates left or up. This next example prints the barcode 2.3 inches up and 0.25 inches right measured from the bottom left corner of the original data.

```
REPLACEDATA=NO
VERPOS=-2.3
HORPOS=+0.25
```

If either VERPOS or HORPOS are specified as zero and the intent is to not move the print position, be sure to include either a plus or minus sign. The following example will move the barcode 2.5 inches to the right and not change the vertical position:

```
VERPOS=+0.0
HORPOS=+2.5
```

The following example, which is the same as the preceding except for removal of a + sign, will print the barcode 2.5 inches to the right and at the top of the page because the 0.0 dimension with no sign is taken as an absolute position:

```
VERPOS=0.0
HORPOS=+2.5
```

PREFIX and **SUFFIX**

```
PREFIX=<character  string>
SUFFIX=<character  string>
```

These statements can be used to add fixed character strings before or after the variable data in the barcode. For example, the following statements will add “JN” (job number) to the beginning of each barcode:

```
PREFIX=JN
```

Prefixes can be very useful in automatically determining in software the particular purpose of each barcode as it is scanned. For example, Job Numbers (JN) can be told apart from Item Numbers (IN) and Shelf Locations (LO).

SYMBOL

```
SYMBOL=<barcode  identifier>
```

This statement defines the type of barcode symbol to be printed. Here is a list of the barcode types supported by BarPump:

CODE39	Code 39, also known as Code 3 of 9
CODE128	Code 128 with automatic switching of subsets
CODE128A	Code 128 Subset A
CODE128B	Code 128 Subset B
CODE128C	Code 128 Subset C
I2OF5	Interleaved 2 of 5
ITF14	Interleaved 2 of 5 UCC Shipping Container Symbol
UCC14	UCC/EAN-128 Shipping Container Symbol
BOOKLAND	EAN Bookland code for ISBN numbers
ISBN	Same as Bookland
CODABAR	Codabar
UPCA	UPC-A retail symbol
EAN13	EAN-13 retail symbol
POSTNET	US Postal Code (5, 9, or 11 digits)
PDF417	2-dimensional PDF-417 code

CHECKSUM

```
CHECKSUM=<type of checksum>
CHECKSUM=<YES/NO>
```

A checksum is optional in some types of barcodes. Code 39, for example, may include a Modulo 43 checksum although it is not commonly used. Interleaved 2 of 5 is often printed with a Modulo 10 check digit. In cases where there is a standard type of checksum associated with the barcode, the specific type of checksum need not be specified. For example, the following statement specifies a checksum for Code 39:

```
CHECKSUM=MOD43
```

Since Modulo 43 is the type of checksum normally used with Code 39, the following statement will produce the same result:

```
CHECKSUM=YES
```

The same statement used with Interleaved 2 of 5 will produce the Modulo 10 checksum typically used with that particular barcode. Most other types of barcodes include a required checksum; this is added automatically by BarPump, so no CHECKSUM statement is needed.

HEIGHT

```
HEIGHT=<height of barcode in inches>
```

The height is the actual vertical dimension of the bars; it does not include the readable text (if used).

NARROWBAR

```
NARROWBAR=<width of narrow bar in inches>
```

The width of a narrow bar is the fundamental dimension that defines the width of a barcode. In general practice, a narrow bar width of less that about 0.009 inches is considered small; a width of less than 0.006 inches is generally too small for most scanners to read reliably. Typical widths are in the range of 0.009 to 0.014 inches; larger dimensions are often used for barcodes that must be read at longer distances.

Note that *BarPump* creates precise barcodes by using the printer's dot-per-inch resolution as the basic unit of measure. If the NARROWBAR dimension does not resolve to a whole number of printer dots, *BarPump* will automatically adjust the dimension. For example, on a 300 dpi printer, 0.009 inches is 2.7 printer dots; BarPump will adjust this to 0.010 inches (3 printer dots).

RATIO

```
RATIO=<wide:narrow ratio>
```

The ratio defines the width of a wide bar relative to the fundamental width of a narrow bar. This setting applies only to Code 39 and Interleaved 2 of 5 barcodes; the ratio in other types of barcodes is fixed. The legal range is from 2.0 (high density) to 3.0 (low density); a ratio of 2.5 is considered medium density. Note that *BarPump* will automatically adjust the ratio if necessary to produce bars which are a whole number of printer dots in width.

SHORTBAR

```
SHORTBAR=<height of a short bar in inches>
```

This statement applies only to Postnet, Planet, and other barcodes which encode data by varying the height of the bars rather than the width. The default height of a short bar is 0.05 inches.

TALLBAR

```
TALLBAR=<height of a tall bar in inches>
```

This statement applies only to Postnet, Planet, and other barcodes which encode data by varying the height of the bars rather than the width. The default height of a tall bar is 0.125 inches.

BARSPERINCH

```
BARSPERINCH=<number of bars per inch>
```

This statement applies only to Postnet, Planet, and other barcodes which encode data by varying the height of the bars rather than the width. Bars are spaced at consistent intervals, specified as the number of bars per inch. The default setting for the Postnet code is 22 bars per inch; the legal range is from 20 to 24 bars per inch.

TEXTPOSITION

```
TEXTPOSITION=<NONE/BELOW/ABOVE>
```

This statement defines whether readable text should be printed above or below the barcode. NONE specifies that the barcode should be printed with no text.

TEXTGAP

```
TEXTGAP=<dimension in inches>
```

When text is printed above or below the barcode, the distance from the bars is calculated automatically based on the size of the text characters. To move the text closer to or farther away from the bars, use TEXTGAP to specify how far the text should be moved from its normal position. A dimension with a plus sign or no sign moves the text down; a minus sign moves the text up.

TEXTALIGN

```
TEXTALIGN=<LEFT/CENTER/RIGHT>
```

This statement specifies whether readable text should be aligned to the left edge of the bars, the right edge of the bars, or centered under the barcode. For example:

```
TEXTALIGN=CENTER
```

TEXTSIZE

```
TEXTSIZE=<size of type in points>
```

The size of readable text is normally set automatically. Use this statement to specifically set the size of the type. Changing the size of the type does not affect the position of the barcode; text printed below the bars grows downward while text printed above the bars grows upward. There are 72 typographic points per inch, so the following statements will print 0.5 inch high (48 point) text centered below the bars:

```
TEXTPOSITION=BELOW
TEXTALIGN=CENTER
TEXTSIZE=48
```

TEXTCPI

```
TEXTCPI=<characters per inch>
```

The spacing of the characters may be set separately from the size (height) of the text. The default is 10 characters per inch.

TEXTEMBED

```
TEXTEMBED=<NO/HALF/FULL>
```

Text may be partially or fully embedded in the barcode, pushed into a notch in the bottom or top of the bars. The default for the UPC-A barcode is to half-embed the text below the bars; the default for Code 39 and Code 128 is not to embed the text.

TEXTFONT

```
TEXTFONT=<identification of font>
```

TEXTFORMAT

```
TEXTFORMAT=<format string>
```

Use this statement to create a template for the readable text. The question mark (?) is used as a placeholder for actual data. For example:

```
TEXTFORMAT=(???) ?????
```

This statement will format the data 12345678 to look like this when it is printed below (or above) the barcode. The data actually encoded in the barcode is not affected.

```
(123) 45678
```

Some types of barcodes have a standard format for readable text, such as the UPC-A symbol or UCC/EAN-128 barcodes.

BEARER_BARS

```
BEARER_BARS=NO
BEARER_BARS=YES
BEARER_BARS=BARS
BEARER_BARS=BOX
```

Bearer bars are heavy lines printed across the top and bottom edges of a barcode. Traditionally used with Interleaved 2 of 5 carton barcodes, the bearer bars prevent erroneous reads when a scanner beam hits the barcode on a diagonal. The bars also provide physical support for the ends of the bars when the symbol is printed with the flexographic process. The ends of the bars may be joined to form a complete box around the barcode.

The default is no bearer bars. Specifying YES or BARS will cause the bars to print; specifying BOX will cause a closed box to print.

ERROR CORRECTION LEVEL (PDF-417 only)

```
ECC=<error correction level>
```

This setting determines how much error correction redundancy will be included in the sym-

bol. The value may range from 0 (minimum correction) to 7 (maximum correction). The higher the number, the more damage a symbol can sustain and still be scannable. Symbols with higher levels of error correction will occupy more space. The default ECC level is 3.

COLUMNS (PDF-417 only)

```
COLUMNS=<number of columns>
```

Data in a PDF-417 symbol is structured in columns and rows. The number of columns defaults to 4; a different number may be specified with the COLUMNS setting. The number of rows required depends on the amount of data, the error correction level, and the number of columns. By adjusting the number of columns it is possible to change the shape of the symbol to fit the space available. A higher number of columns will result in a low and wide symbol; as the number of columns is reduced, the symbol will become taller and narrower.